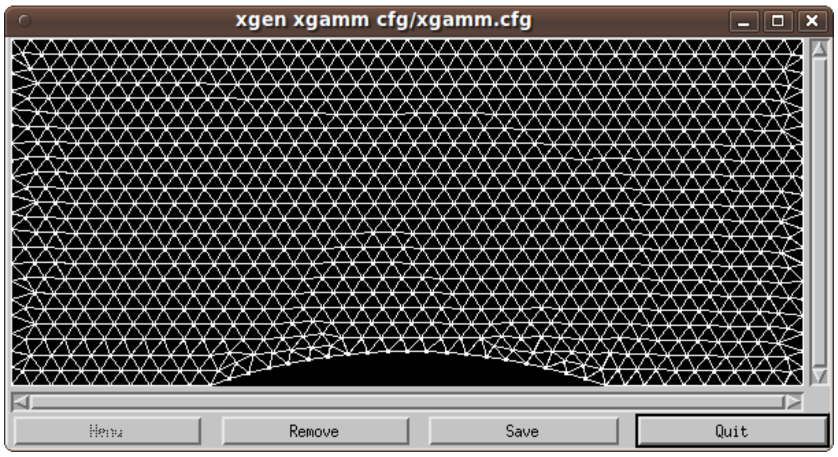
XGEN

A simple generator of uniform unstructured triangular meshes.



Version 6.0

Email: hpfem-group@googlegroups.com. Home page: http://hpfem.org.

1 Download, Installation, and Testing

A few external libraries are required to compile XGEN. In Ubuntu, install them using:

```
sudo apt-get install libmotif-dev libmotif3 lesstif2 x11proto-print-dev
```

In the next step, clone the Git repository:

```
git clone http://github.com/hpfem/xgen.git
```

After that, change dir to xgen/ and build XGEN by typing “make”. The present Makefile works for Ubuntu, but it should work on other Linux and Unix platforms with no or minor modifications. Finally, check that XGEN works properly by typing

```
./xgen xgamm cfg/xgamm.cfg
```

After this, a graphical window with the geometry from the front page should appear. Press the ”Mesh” button to create a mesh.

2 Introduction

XGEN is a simple but very robust generator of unstructured triangular meshes. It can operate both in interactive and batch modes. The geometry is entered as a positively (counter clock-wise) oriented polygon that consists of straight line segments and/or circular arcs. Holes must be oriented clock-wise. If circular arcs are present, then the resulting mesh will contain curved elements.

2.1 Interactive mode

After the domain is defined, XGEN calculates its area and throws inside an appropriate number of random points (Fig. 1).

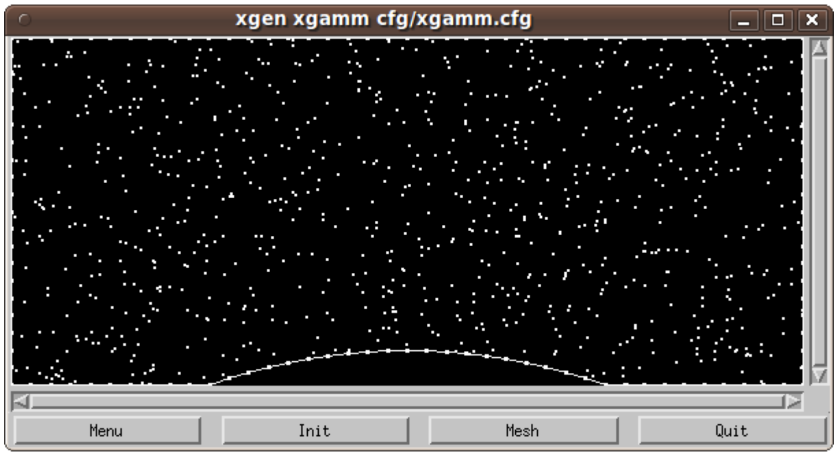


Figure 1: Random points.

These points are then relaxed using an "electrostatic" simulation until an equilibrium is reached (Fig. 2).

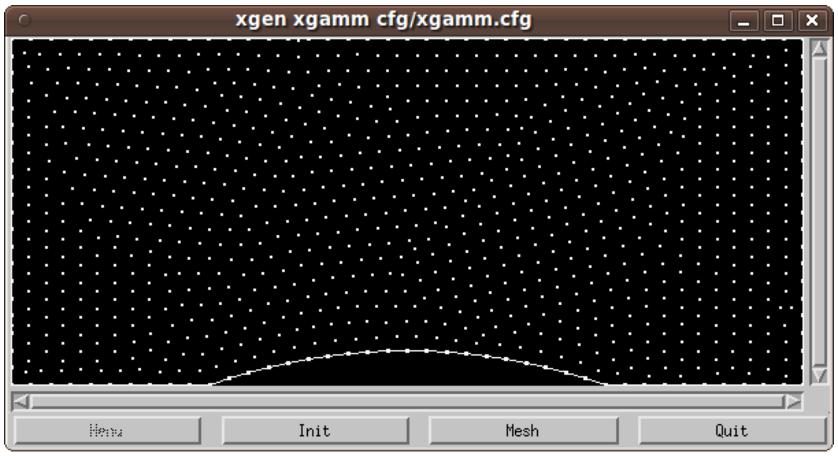


Figure 2: An "electrostatic" simulation is run to reach a local equilibrium.

Alternatively, the user can choose a regular overlay pattern by calling XGEN with the "-overlay" option:

```
./xgen xgamm cfg/xgamm.cfg -overlay
```

In interactive mode, the same can be achieved through the Init → Overlay buttons. The overlay pattern is illustrated in Fig. 3.

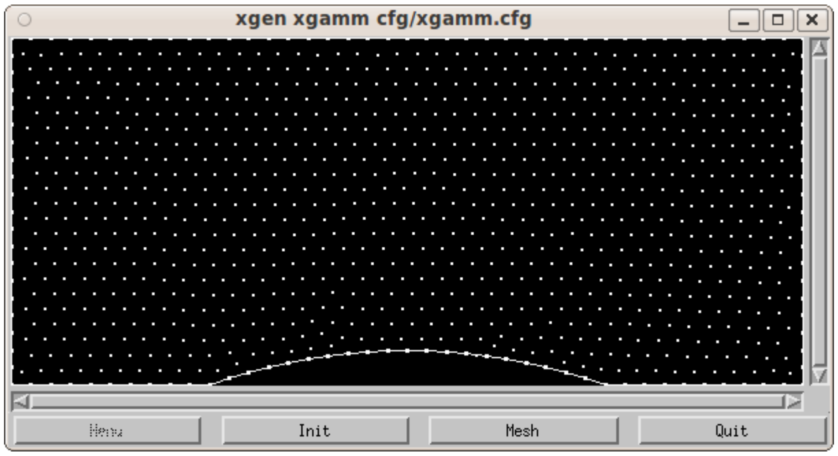


Figure 3: Overlay points.

In interactive mode, the user can add and remove points using the left and middle mouse buttons (left and right on a two-button mouse). Mesh can be generated at any time by pressing the "Mesh" button. The "Menu" button launches a menu with several self-explanatory functions (Fig. 4).

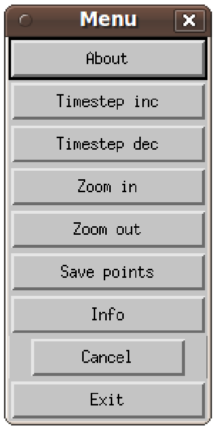


Figure 4: XGEN menu.

2.2 Batch mode

XGEN can be run in batch mode using the option "-nogui N" where N is the number of smoothing iterations over grid points (time steps of the electrostatic simulation). For example,

```
./xgen xgamm cfg/xgamm.cfg -nogui 100 -overlay
```

will use an initial uniform overlay pattern of grid points and perform 100 smoothing steps. In batch mode, the mesh is saved to the file "out.mesh".

3 Defining Geometry via Segments

The simplest way to enter a geometry is to prepare a text file that defines the boundary as a positively oriented list of straight lines and/or circular segments. Each segment is entered using 5 numbers: the boundary marker, starting point coordinates, subdivision, and angle. If the angle is zero, the segment is a straight line. Before entering a hole, one needs close the current boundary component using the '=' symbol. The following file describes a square $(0,3)^2$ with a circular hole of radius $\sqrt{2}$ in its middle.

```
# This is an XGEN cfg file to the class 'xlist'.
# Edges list:
# (boundary marker, x_start, y_start, subdivision, angle)
1 0.0 0.0 10 0.0
2 3.0 0.0 10 0.0
3 3.0 3.0 10 0.0
4 0.0 3.0 10 0.0

# Symbol introducing new component.
=

# Holes must be negatively oriented.
5 1.0 1.0 3 -90.0
5 1.0 2.0 3 -90.0
5 2.0 2.0 3 -90.0
5 2.0 1.0 3 -90.0
```

You can find this file in the repository under `cfg/xlist_hole.cfg`. XGEN is then called as follows,

```
./xgen xlist cfg/xlist_hole.cfg
```

The resulting mesh is shown in Fig. 5.

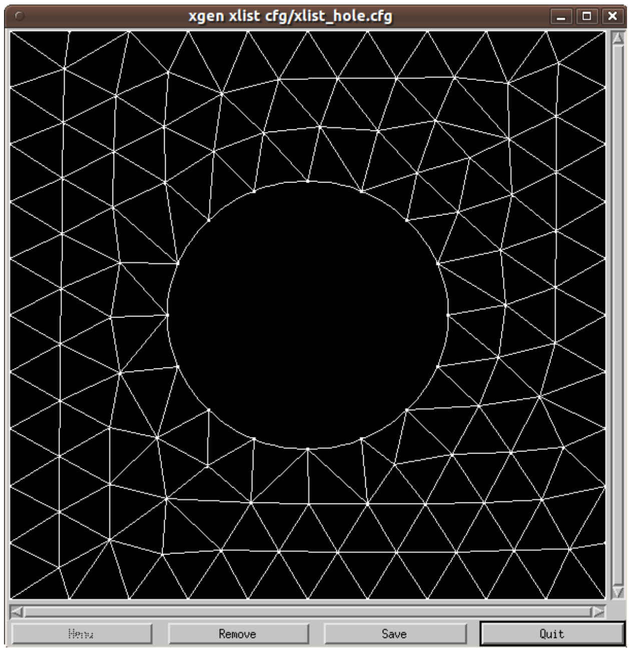


Figure 5: Sample geometry corresponding to the above input file.

4 Creating Custom Projects

In the previous section we saw how to enter the geometry via a positively oriented list of lines and/or circular arcs. In fact, this was just one specific application (called ”xlist”) that consists of less than 50 lines of code (incl. comments).

To create a custom project such as ”xlist”, one needs to define a descendant of a basic class Xgen. Let us paste the code of the class ”xlist” here for illustration:

```
/*
 *   class xlist:
 *   general polygonal domain with possibly curved edges
 *   see cfg/xlist_curved_6.cfg for an example of input file
 */

class xlist: public Xgen {
  public:
  xlist(char *cfg_filename, bool nogui, int nsteps, bool overlay)
```

```
        : Xgen(nogui, nsteps, overlay) {XgInit(cfg_filename);}
  virtual void XgReadData(FILE *f);
};

// Boundary edges have the following format:
// <marker start_x, start_y, subdivision, angle>
// where angle = 0 for straight edges and angle != 0 for curved.
// New component is introduced with '='.
void xlist::XgReadData(FILE *f) {
  int marker, subdiv, end = 0;
  Point a, b, c;
  double angle;
  char test[20];

  while(Get(f, &marker)) {
    if(!Get(f, &a.x)) XgError("Couldn't read a starting point.");
    if(!Get(f, &a.y)) XgError("Couldn't read a starting point.");
    if(!Get(f, &subdiv)) XgError("Couldn't read a subdivision number.");
    if(!Get(f, &angle)) XgError("Couldn't read an angle.");

    // Closing a boundary loop.
    XgCreateNewBoundaryComponent();

    XgAddBoundarySegment(marker, a.x, a.y, subdiv, angle);
    c = a;
    while(end = !Get(f, test), (end || test[0] == '=') ? false : true) {
      marker = atoi(test);
      if(!Get(f, &b.x)) XgError("Couldn't read a starting point.");
      if(!Get(f, &b.y)) XgError("Couldn't read a starting point.");
      if(!Get(f, &subdiv)) XgError("Couldn't read a subdivision number.");
      if(!Get(f, &angle)) XgError("Couldn't read an angle.");
      XgAddBoundarySegment(marker, b.x, b.y, subdiv, angle);
      a = b;
    }
  }
}
```

This code can be found in the file `src/main.cpp` starting with line 199. The method `XgReadData()` reads data from a user-supplied text file and uses an elementary function

```
  XgAddBoundarySegment(marker, a.x, a.y, subdiv, alpha);
```

to create an arbitrary geometry. The function `XgAddBoundarySegment()` adds a new segment with boundary marker ”marker”, starting point ”a”, subdivision ”subdiv” and angle ”alpha”. New boundary component (loop) is created with

```
  XgCreateNewBoundaryComponent();
```

After creating a new class, add the following three lines into the main() function of the file src/main.cpp, to make sure that the class' name can be used as a command-line parameter:

```
  if(!strcmp(argv[1], "xlist")) {
    xlist X(argv[2]);
    XgMainLoop(&X, argc, argv);
  }
```

That's it!

5 Mesh Output Format

By default, XGEN creates meshes in Hermes2D native format (http://hpfem.org/hermes). This can be changed by overriding the virtual method `XgUserOutput()`. The mesh with the circular hole generated in the previous section can be found in `meshes/xlist_hole.mesh`. The mesh file contains four parts: `vertices`, `elements`, `boundaries`, and `curves`:

```
# XGEN mesh in Hermes2D format
# Project: cfg/xlist_hole
# Edges are positively oriented

# Vertices:
vertices =
{
  { 0, 0 },
  { 0.3, 0 },
  { 0.6, 0 },
  .
  .
  .
  { 0.228144, 2.8167 },
  { 0.165974, 0.244247 },
  { 0.247855, 0.761208 }
}

# Elements:
elements =
{
  { 40, 41, 63, 0 },
  { 41, 42, 75, 0 },
  { 42, 43, 68, 0 },
  .
  .
  .
  { 89, 59, 94, 0 },
  { 68, 109, 42, 0 },
  { 42, 109, 75, 0 }
}
```

The first three indices refer to the vertex array. The last number on each line is a material marker. XGEN currently does not allow for different material markers, and thus a default value "0" is used for all triangles.

```
# Boundary markers:
# (bdy_vertex_1 bdy_vertex_2 edge_index)
boundaries =
{
  { 0, 1, 1 },
  { 1, 2, 1 },
  { 2, 3, 1 },
  .
  .
  .
  { 53, 54, 5 },
  { 54, 55, 5 },
  { 55, 40, 5 }
}
```

The first two indices refer to the vertex array, and the last number on each line is the material marker.

```
# Circular arcs:
# (bdy_vertex_1 bdy_vertex_2 central_angle)
curves =
{
  { 40, 41, -22.5 },
  { 41, 42, -22.5 },
  { 42, 43, -22.5 },
  .
  .
  .
  { 53, 54, -22.5 },
  { 54, 55, -22.5 },
  { 55, 40, -22.5 }
}
```

The ”curves” section is empty if the boundary is formed by straight lines only. For each curved element edge, the first two numbers are indices refering to the vertex array, and the last one is a central angle of the circular arc.

6 Saving and Loading Points

Grid points can be saved using the button “Save points” in ”Menu”. To load points, use buttons “Init” and “Load”. The output file only contains points coordinates as two numbers per line. While loading, points that are checked to be out of the domain are ignored.

7 XGEN Configuration File

Upon execution, XGEN attempts to open the file .XgenConfig in the current directory. This file can be provided by the user and it contains the following defaults:

```
# Path to directory with point sets:
points/
# Path to directory with meshes:
meshes/
# SubLoopsNumber:
100
# Initial size limit:
600
# Boundary redraw interval:
200
# Time step change ratio:
0.75
# Zoom ratio:
0.95
```

Variables that are not read from this file are set to some default values. Here the points and mesh file paths specify where the points and mesh files are stored, respectively. SubLoopsNumber is the number of points to be moved before returning control to the Window Manager. The next value is the size of the window when placed on the screen. The shape of the window is determined according to the shape of the triangulated domain by the generator itself. BoundaryRedrawInterval is the number of points to be moved before the boundary is redrawn. TimestepConst is a number between 0 and 1 that defines the change of the time-step after pressing “Timestep inc” and “Timestep dec” in “Menu”. The last variable ZoomRatio has a similar meaning in context of zooming.

8 Visualizing Meshes with Hermes2D

The easiest way to visualize an XGEN mesh is to copy the mesh file into the directory ”.../hermes/hermes2d/tutorial/01-mesh” in the Hermes repository as "domain.mesh", comment out local refinements in the main.cpp file, build the example again typing "make" in its directory, and run "mesh". Then click into the graphics window containing the mesh, and press 's' – this will save a BMP image on the disk. An example for the above GAMM channel project is shown in Fig. 6.

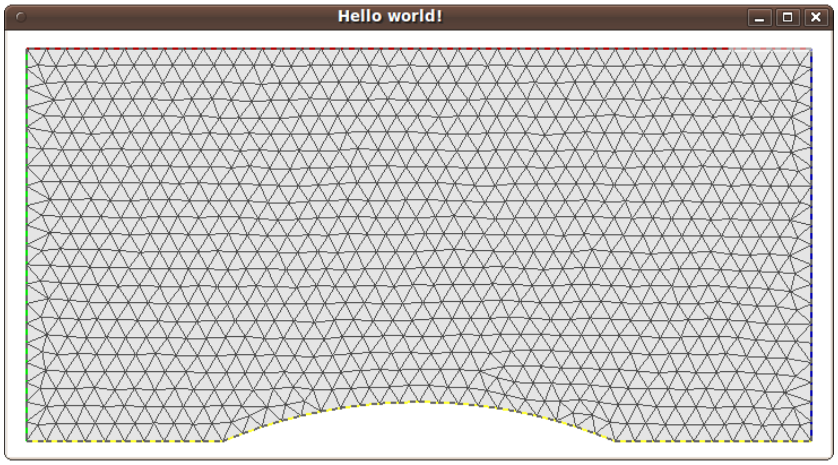


Figure 6: XGEN mesh visualized in Hermes2D.

9 Additional examples

The file src/main.cpp contains a few more sample descendants which can be tried via:

```
./xgen xsquare cfg/xsquare.cfg &
./xgen xhole cfg/xhole.cfg &
./xgen xcirc cfg/xcirc.cfg &
./xgen xstep cfg/xstep.cfg &
```

Pavel Solin, May 1995 (updated December 2010)